Labeled directed graphs and FSA as classifiers of strings

Andrei Kelarev, Byeong Kang, Arthur Sale, Ray Williams

School of Computing
University of Tasmania
Private Bag 100, Hobart
Tasmania 7001, Australia

Abstract

The classical concept of a finite state automaton (FSA) has long been used in computer science to classify formal languages regarded as sets of strings. This paper surveys and unifies recent results on strings recognized by FSA defined by labeled directed graphs, and highlights several interesting open questions. In particular, one of our open questions deals with a new method of applying a related construction to classification of strings.

1 Introduction

Labeled graphs and diverse objects derived from them are most important essential tools of computer science. Various applications of labeled graphs have been actively investigated by many researchers. We refer to [13] for a dynamic survey on graph labeling available online from the Electronic Journal of Combinatorics. Many interesting results concerning graph labelings have appeared in the literature recently, see [2, 3, 4, 29, 30, 32, 33, 40, 43, 44, 45, 47, 48].

The first aim of this paper is to survey and unify recent results devoted to the investigation of combinatorial properties of strings recognized by FSA defined by labeled directed graphs. Secondly, we raise and highlight several interesting open questions. In particular, one of our open questions deals with a new method of applying a related construction to classification of strings.

[1]Research supported by IRGS grant K14313 of the University of Tasmania and Discovery grant DP0449469 from the Australian Research Council.
Email: *Andrei.Kelarev@utas.edu.au* URL: `www.comp.utas.edu.au/users/kelarev/`
Email: *bhkang@utas.edu.au* URL: `www.comp.utas.edu.au/users/bhkang/`
Email: *Arthur.Sale@utas.edu.au* URL: `www.comp.utas.edu.au/users/ahjs/`
Email: *R.Williams@utas.edu.au* URL: `www.comp.utas.edu.au/users/rwilliams/`

2 Preliminaries

We use standard concepts on algorithms and graphs, following [9] and [42]. Likewise, for the background terminology and notation of automata and languages theory we refer the readers to [12, 36, 39, 50]. As it is customary in the literature, we say that a question is *decidable* if there exists a combinatorial algorithm which finds the answer.

Throughout the word *graph* means a finite directed graph without multiple edges but possibly with loops. Let X be an alphabet, i.e., a finite set of elements called letters. A *string* over X is a finite ordered sequence $x_1, \ldots, x_n$, where $x_1, \ldots, x_n \in X$. The set of all (non-empty) strings over X is denoted by X^+. It is often convenient to include into consideration the empty string without letters. It will be denoted by 1. The set $X^* = X^+ \cup \{1\}$ of all strings over X is called the *free monoid* generated by X, see [12].

Classification of data is important in data mining, see [49] and also, for example, [10], [11], [28], [37]. One of the standard well-known methods of classifying strings uses the notion of a finite state automaton (FSA). As a binary classifier, each FSA accepts some strings and rejects others. The set of all strings accepted by an FSA is called the *language* recognized by the automaton, see [12].

Several new ways of defining classical finite state classifiers have been considered in the literature, and a general concept based on graph labelings has been introduced in [21]. Now it is possible to unify different results and determine how properties of the classifiers depend on the properties of the original graph labeling.

Let $D = (V, E)$ be a graph, $\ell : X \to \{+, -\}$ and $f : X \to V$ any mappings, and let T be a subset of $V \cup \{1\}$. The finite state automaton $\mathrm{Atm}(D, T, f, \ell)$ of the graph D is the finite state binary classifier with

(DA1) the set of states $V \cup \{1\}$;

(DA2) the initial state 1;

(DA3) the set of terminal states T;

(DA4) the next-state function given by

$$a \cdot x = \begin{cases} f(x) & \text{if } \ell(x) = + \text{ and } (a, f(x)) \in E, \text{ or if } a = 1, \\ a & \text{if } \ell(x) = - \text{ and } (f(x), a) \in E, \end{cases}$$

for a state a and $x \in X$.

Thus the edges of the transition diagram of $\mathrm{Atm}(D, T, f, \ell)$ are also edges or reversed edges of the graph D.

Let $\mathcal{D}$ be a class of graphs and let $X = X_+ \dot{\cup} X_-$ be an alphabet. Denote by $\mathcal{G}(\mathcal{D}, X_+, X_-)$ the class of sets of strings recognized by automata $\mathrm{Atm}(D, T, f, \ell)$ of graphs of class $\mathcal{D}$ over the alphabet X with the given labeling $\ell(X_+) = \{+\}$ and $\ell(X_-) = \{-\}$. In the case when $\mathcal{D}$ is the class of all directed graphs we denote this class of sets of strings by $\mathcal{G} = \mathcal{G}(X_+, X_-)$. The sets of strings recognized by automata $\mathrm{Atm}(D, T, f, \ell)$ have interesting combinatorial properties.

3 Combinatorial properties

This section is devoted to combinatorial properties of sets of strings recognized by the finite state automata $\mathrm{Atm}(D, T, f, \ell)$ of graphs.

Theorem 1 ([21]) *For every set L of strings over an alphabet X, the following conditions are equivalent:*

(i) *there exists a directed graph D and a finite state automaton $\mathrm{Atm}(D, T, f, \ell)$ such that L is recognized by this automaton;*

(ii) *there exist two disjoint subsets X_- and X_+ of X such that $X = X_- \dot\cup X_+$ and, for all $x \in X_+$, $y \in X_-$, $z \in X$ and $u, v \in X^*$, the following implications hold:*

(a) *$zxu \in L$ implies $xu \in L$,*

(b) *$xu, zxv \in L$ implies $zxu \in L$,*

(c) *$zyu \in L$ implies $zy^*u \in L$,*

(d) *$zv, zyu \in L$ implies $zyv \in L$,*

(e) *$xyu \in L$ if and only if $yxu \in L$.*

A detailed description of all sets of strings recognized by automata $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{+\}$ is given by the following theorem in terms of combinatorial properties of strings.

Theorem 2 ([22]) *For any language L over an alphabet X, the following conditions are equivalent:*

(i) *L is recognized by a finite state automaton $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{+\}$;*

(ii) *at least one of the following two conditions is satisfied for all $x, y \in X$, and $u, v \in X^*$:*

(a) *$xyu \in L$ implies $yu \in L$, and*

$xu, yxv \in L$ implies $yxu \in L$;

(b) *$yu \in L$ implies $xyu \in L$, and*

$yxu \in L$ implies $xu \in L$ or $yxv \in L$.

4 Regular expressions

This section deals with descriptions of sets of strings recognized by finite state automata $\mathrm{Atm}(D, T, f, \ell)$ of graphs in terms of regular expressions for these sets or their complements. Given a relation $R \subseteq X \times X$, put

$$R^{-1} = \{(x, y) \mid (y, x) \in R\}.$$

Theorem 3 ([21]) *For every set L of strings over an alphabet X, the following conditions are equivalent:*

(i) *there exists a directed graph D and a finite state automaton $\mathrm{Atm}(D, T, f, \ell)$ such that L is recognized by this automaton;*

(ii) *there exist a subset X_T of X, disjoint subsets X_- and X_+ of X, and relations $R_1 \subseteq X_+ \times X_+$, $R_2 \subseteq X_- \times X_-$, and $R_3 \subseteq X_- \times X_+$ such that $X = X_- \dot{\cup} X_+$ and the set $X^+ \setminus L$ of strings has the following regular expression*

$$(X_N \cap X_-)X_-^* + X^*(X_N \cap X_+)X_-^* + \sum_{(x_i,x_j)\in R_1 \cup R_3^{-1}} X^* x_i X_-^* x_j X^* + \sum_{(x_i,x_j)\in R_2 \cup R_3} x_i X_-^* x_j X^*, \tag{1}$$

where X_N stands for $X \setminus X_T$.

Note that a set L of strings is accepted by $\mathrm{Atm}(D, T, f, \ell)$ if and only if $L \setminus \{1\}$ is accepted by $\mathrm{Atm}(D, T \setminus \{1\}, f, \ell)$.

A detailed description of all sets of strings recognized by automata $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{+\}$ is given by the following theorem in terms of regular expressions.

Theorem 4 ([22]) *For any language L over an alphabet X, the following conditions are equivalent:*

(i) *L is recognized by a finite state automaton $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{+\}$;*

(ii) *there exist disjoint subsets X_1, X_2 of X and a relation $R \subseteq X \times X$ such that the language $L \setminus \{1\}$ or $X^+ \setminus L$ has the following regular expression*

$$X^* X_1 X^* + X^* X_2 + \sum_{(x_j,x_i)\in R} X^* x_i x_j X^*. \tag{2}$$

Theorem 5 ([22]) *It is decidable whether a regular language belongs to the class $\mathcal{G}$.*

Denote by $\mathcal{G}_a$ the subclass of $\mathcal{G}$ containing the sets of strings over X satisfying condition (a) of Theorem 4 and by $\mathcal{G}_b$ the subclass of $\mathcal{G}$ containing the sets of strings over X satisfying condition (b). Let $\tilde{\mathcal{G}}_i = \{L \subseteq X^* \mid \overline{L} = X^* \setminus L \in \mathcal{G}_i\}$, for $i = a, b$.

Theorem 6 ([22]) *The classes $\mathcal{G}_a$ and $\mathcal{G}_b$ satisfy equalities $\tilde{\mathcal{G}}_a = \mathcal{G}_b$ and $\tilde{\mathcal{G}}_b = \mathcal{G}_a$. A set L of strings belongs to the class $\mathcal{G}_a \cap \mathcal{G}_b$ if and only if there exists a subset X_2 of X such that $L \setminus \{1\}$ is given by the regular expression $X^* X_2$.*

In particular, the class $\mathcal{G}_a \cap \mathcal{G}_b$ contains the sets of strings $\emptyset$, $\{1\}$, X^+, and X^*.

Theorem 7 ([22]) *The class $\mathcal{G}_a$ contains all regular sets of strings given by the regular expressions of the form*

$$X_1 X_2^* X_3 X_4^* \cdots X_{2n}^* X_{2n+1}, \tag{3}$$

where $X_1, X_2, \ldots, X_{2n+1} \subseteq X \cup \{1\}$, and $X_i \cap X_j = \{1\}$, for all $1 \leq i < j \leq 2n+1$.

It is easily seen that all sets of strings in the class $\mathcal{G}_b$, except $\emptyset$ and $\{1\}$, are infinite. On the other hand, the class $\mathcal{G}_a$ contains some finite sets of strings, but not all, as the following theorem shows.

Theorem 8 ([22]) *Let L be a finite language over an alphabet X, where $|X| = n$. If $L \in \mathcal{G}_a$, then $|L| \leq 2^n$.*

It is well-known that all finite sets of strings are regular. Hence we see that many regular sets of strings are not recognized by finite state automata $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{+\}$.

Proposition 1 ([22]) *There exists a set of strings which belongs to $\mathcal{G}_a$ but cannot be defined by a regular expression of the form (3).*

We say that a language on X is *trivial* if it is equal to $\emptyset$ or X^+. Obviously, trivial languages can be recognized by automata $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{-\}$. The next theorem characterizes all nontrivial languages recognized by automata $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{-\}$.

Theorem 9 ([24]) *A nontrivial language $L \subseteq X^+$ is recognized by the automaton $\mathrm{Atm}(D, T, f, \ell)$ of some graph with $\mathrm{Im}(f) = \{-\}$ if and only if the letters of X can be reordered and denoted by*

$$\begin{aligned} X \;=\; \{&x_{11}, \ldots, x_{1k_1}, \ldots, x_{\ell 1}, \ldots, x_{\ell k_\ell}, y_{11}, \ldots, y_{1m_1}, \ldots, y_{\ell 1}, \ldots, y_{\ell m_\ell}, \\ &z_{11}, \ldots, z_{1n_1}, \ldots, z_{\ell 1}, \ldots, z_{\ell n_\ell}, u_1, \ldots, u_p, w_1, \ldots, w_t\}, \end{aligned}$$

so that either L or $X^+ \setminus L$ has the following regular expression

$$\begin{aligned} &(u_1 + \cdots + u_p)+ \\ &\quad + \quad (x_{11} + \cdots + x_{1k_1} + y_{11} + \cdots + y_{1m_1})(x_{11} + \cdots + x_{1k_1} + z_{11} + \cdots + z_{1n_1})^* \\ &\quad + \quad \cdots \\ &\quad + \quad (x_{\ell 1} + \cdots + x_{\ell k_\ell} + y_{\ell 1} + \cdots + y_{\ell m_1})(x_{\ell 1} + \cdots + x_{\ell k_\ell} + z_{\ell 1}, \ldots, z_{\ell n_\ell})^*, \end{aligned}$$

where $p, \ell, t, k_1, \ldots, k_\ell, n_1, \ldots, n_\ell, m_1, \ldots, m_\ell \geq 0; \ell + p \geq 1$ and if $k_i = 0$, then $m_i \neq 0$ and $n_i = 0$.

5 Grammars

A complete description of all grammars generating sets of strings recognized by automata $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{+\}$ is given by the following theorem.

Theorem 10 ([22]) *For any language L over an alphabet X, the following conditions are equivalent:*

(i) *L is recognized by a finite state automaton $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{+\}$;*

(ii) *there exist subsets $Q \subseteq X$ and $P \subseteq X \times X$ such that the language $L \setminus \{1\}$ or $X^+ \setminus L$ is generated by the right linear grammar with the alphabet X, the set $W = \{x' \mid x \in X\} \cup \{s_0\}$ of nonterminal symbols, the start symbol s_0, and productions*

$$\begin{array}{lcll} s_0 & \to & xx' & \text{for all } x \in X, \\ x' & \to & yy' & \text{for all } (y, x) \in P, \\ x' & \to & 1 & \text{for all } x \in Q. \end{array} \tag{4}$$

6 Closure properties

Closure properties of sets of strings play one of the central roles in solutions to various problems. It has been established that $\mathcal{G}$ contains certain fairly large known subclasses, is a proper subclass of the class of sets of strings recognizable by finite state automata, and is closed under the Kleene $*$-operation and complement. Although the whole class $\mathcal{G}$ is not closed for union, intersection, and product, it has been represented as a union of two subclasses $\mathcal{G}_a$ and $\mathcal{G}_b$ such that $\mathcal{G}_a$ is closed under intersection and left derivative, and $\mathcal{G}_b$ is closed under union and right derivative.

Theorem 11 ([22]) *The class of sets of strings recognized by finite state automata $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{+\}$ is not closed under union, intersection, or product.*

For any language L and string $u \in X^*$, let $Lu^{-1} = \{w \in X^* \mid wu \in L\}$ and $u^{-1}L = \{w \in X^* \mid uw \in L\}$. A class $\mathcal{L}$ of sets of strings is said to be *closed under left (right) derivative* if $L \in \mathcal{L}$ implies $u^{-1}L \in \mathcal{L}$ (respectively, $Lu^{-1} \in \mathcal{L}$).

Theorem 12 ([22]) *The class $\mathcal{G}$ is closed under complement, $\mathcal{G}_a$ is closed under intersections and left derivative, and $\mathcal{G}_b$ is closed under union and right derivative.*

Theorem 13 ([22]) *The classes $\mathcal{G}_a$ and $\mathcal{G}_b$ are closed under the Kleene $*$-operation.*

Theorem 14 ([20]) *For every labeling $X = X_+ \cup X_-$ of the alphabet, the class $\mathcal{G} = \mathcal{G}(X_+, X_-)$ is closed under intersection.*

Theorem 15 ([20]) *The class $\mathcal{G}(X_+, \emptyset)$ is closed under the Kleene $*$-operation, but the class $\mathcal{G}(\emptyset, X_-)$ is not.*

Theorem 16 ([20]) *Let $\mathcal{D}$ be any class consisting of complete graphs. Then the following conditions are equivalent:*

(i) $L \in \mathcal{G}(\mathcal{D}, X_+, X_-)$*;*

(ii) *there exist subsets $Y_1 \subseteq X_+$ and $Y_2 \subseteq X_-$ such that $X^+ \setminus L$ has the following regular expression*

$$Y_2 X_-^* + X^* Y_1 X_-^* \tag{5}$$

(iii) *there exist subsets $Y_1 \subseteq X_+$ and $Y_2 \subseteq X_-$ such that L has a regular expression of the form* (5) *or* (6)*:*

$$1 + Y_2 X_-^* + X^* Y_1 X_-^* \tag{6}$$

Theorem 17 ([20]) *Let $\mathcal{D}$ be any class consisting of complete graphs. Then the class $\mathcal{G}(\mathcal{D}, X_+, X_-)$ is closed under intersection, union and the Kleene $*$-operation.*

The *in-neighbourhood* and the *out-neighbourhood* of a vertex u of a graph $D = (V, E)$ are, respectively, the sets

$$\mathrm{In}(u) = \{w \in V \mid (w, u) \in E\} \quad \text{and} \quad \mathrm{Out}(u) = \{w \in V \mid (u, w) \in E\}.$$

Denote by K_n the complete graph with n vertices and all loops. A *null* graph N_k is a graph on k vertices without edges. Let $K'_{1,1}$ be the graph with the set $\{a, b\}$ of vertices and the set $\{(a, a), (b, a)\}$ of edges. Put $K_{1,1} = K'_{1,1} + (b, b)$. Let $\mathcal{K}$ be the set of the four graphs on the same set $\{a, b, c\}$ of vertices such that each graph of $\mathcal{K}$ has the edges $(a, a), (b, b), (b, a), (b, c)$, it may contain the loop (c, c), and it may simultaneously contain the edges (a, b) and (a, c).

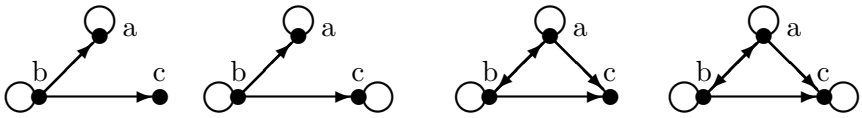

Figure 1: The set $\mathcal{K}$

Theorem 18 ([20]) *Let $|X| \geq 3$. The class $\mathcal{G}(\mathcal{D}, \emptyset, X_-)$ is closed under the Kleene $*$-operation if and only if one of the following conditions is satisfied:*

(1) *not all graphs in $\mathcal{D}$ are null, and each graph of $\mathcal{D}$ either is isomorphic to one of the graphs K_2, $K_{1,1}$, $K'_{1,1}$, or is a disjoint union of some copies of K_1 and N_1;*

(2) *the following two conditions hold:*

(i) *for every graph $D = (V, E)$ of $\mathcal{D}$, the out-neighbourhood $\mathrm{Out}(v)$ of each vertex $v \in V$, is equal to one of the sets $V, \{v\}$, or $\emptyset$;*

(ii) *the class $\mathcal{D}$ contains a graph with a subgraph isomorphic to one of graphs of $\mathcal{K}$.*

7 Combinatorial algorithms

There are three major methods of transforming regular expressions into finite automata, which can be used in the algorithms of this section (see [51], §3.2). Similarly, the computation of a minimal automaton can be carried out in several ways. For instance, the *reduction algorithm* due to Moore starts with a given automaton and computes successive approximations of the Nerode equivalence (see, for example, [5]). A careful implementation of this algorithm has been proposed by Hopcroft [1], who proves that it can be carried out in time $O(N \log N)$ for N-state automaton. Let us begin with a general result.

Theorem 19 ([21]) *It is decidable whether a regular language is recognizable by finite state automata* $\mathrm{Atm}(D, T, f, \ell)$ *of graphs.*

The first algorithm of this section verifies whether a set of strings defined by a regular expression is recognized by an automaton $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{-\}$.

Algorithm 1

// Input: A regular expression R defining a language L.

// Output: Minimal automaton $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{-\}$ recognizing this language, if it exists.

Step 1. Find an automaton $\mathcal{A}$ recognizing L.

Step 2. Reduce $\mathcal{A}$ and find an equivalent minimal automaton $\mathcal{M}$.

Step 3. If $\mathcal{M}$ recognizes a nontrivial language, then check whether $\mathcal{M}$ is of the form shown in Figures 2 or 3 up to notation of letters of the alphabet. If not, then the language is not recognised by automata $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{-\}$, and we stop. Otherwise, we go to the next step.

Step 4. In order to construct a graph for the required automaton $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{-\}$, we use parameters $p, \ell, t,\ k_1, \ldots, k_\ell,\ n_1, \ldots, n_\ell,\ m_1, \ldots, m_\ell$ explained in Figure 2. We may assume that these numbers are ordered so that $m_1 = \cdots = m_{r_2} = 0$, $m_{r_2+1}, \ldots, m_\ell \geq 1$, $n_1 = \cdots = n_{r_1} = n_{r_2+1} = \cdots = n_{r_3} = 0$ and $n_{r_1+1}, \ldots, n_{r_2}$, $n_{r_3+1}, \ldots, n_\ell \geq 1$, for some $1 \geq r_1 \geq r_2 \geq r_3 \geq \ell$. We define all connected components of the required graph D using the numbers p, r_1, r_2 and r_3. If $p > 0$, then one of the connected components of the required graph D is an isolated vertex c_0 without edges. The graph D has r_1 connected components isomorphic to K_1, $r_2 - r_1$ connected components isomorphic to K_2, $r_3 - r_2$ connected components isomorphic to K_1^1 and $\ell - r_3$ connected components isomorphic to K_2^1. Then the language L is recognized by the automaton $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{-\}$ of the graph D.

The next algorithm verifies whether a regular language is recognized by the automaton $\mathrm{Atm}(D, T, f, \ell)$ of an undirected graph with $\mathrm{Im}(f) = \{-\}$.

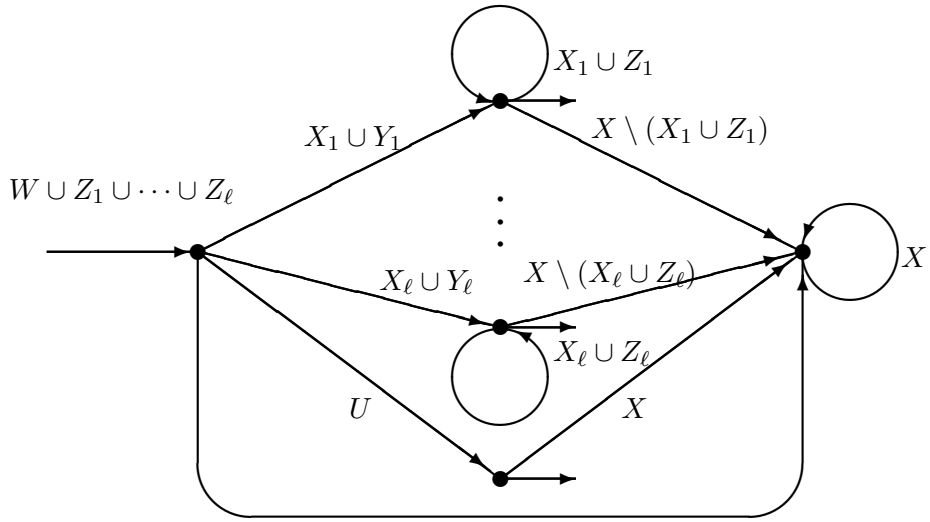

Figure 2: Alphabet $X = \bigcup_{i=1}^{\ell}(X_i \cup Y_i \cup Z_i) \cup U \cup W$, where $X_i = \{x_{i1}, \ldots, x_{ik_i}\}, Y_i = \{y_{i1}, \ldots, y_{im_i}\}, Z_i = \{z_{i1}, \ldots, z_{in_i}\}, U = \{u_1, \ldots, u_p\}, W = \{w_1, \ldots, w_t\}$, $p, \ell, t, k_1, \ldots, k_\ell, n_1, \ldots, n_\ell, m_1, \ldots, m_\ell \geq 0$; $\ell + p \geq 1$ and if $k_i = 0$, then $m_i \neq 0$ and $n_i = 0$.

Algorithm 2

// Input: A regular expression R defining a language L.

// Output: Undirected graph with a minimal automaton $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{-\}$ recognizing this language, if it exists.

Step 1. Find an automaton $\mathcal{A}$ recognizing L.

Step 2. Reduce $\mathcal{A}$ and find an equivalent minimal automaton $\mathcal{M}$.

Step 3. If $\mathcal{M}$ recognizes a nontrivial language, then check whether $\mathcal{M}$ is of the form shown in Figures 4 or 5 up to notation of letters of the alphabet. (Note that these automata are simpler than those on Figures 2 and 3). If $\mathcal{M}$ is not of this form, then the language is not recognised by automata $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{-\}$ of undirected graph, and we stop. Otherwise, go to the next step.

Step 4. In order to define a graph D with $\mathrm{Im}(f) = \{-\}$ and $\mathrm{Atm}(D, T, f, \ell)$ recognising L, we use notation p, ℓ, t, $n_1, \ldots, n_\ell$, $k_1, \ldots, k_\ell$ explained in Figure 4. We may assume that these numbers are ordered so that $n_1, \ldots, n_r = 0$ and $n_{r+1}, \ldots, n_\ell \geq 1$ for some r. Let us define all connected components of the graph D. If $p > 0$, then one connected component of D is an isolated vertex without edges. The graph D also has r connected components isomorphic to K_1, and $\ell - r$ connected components isomorphic to K_2. Then it follows from Theorem 9 that the language L is recognized by the automaton $\mathrm{Atm}(D, T, f, \ell)$ with $\mathrm{Im}(f) = \{-\}$ of the undirected graph D.

For $k \geq 1$ and $u \in X^+$ such that $|u| \geq k$, denote by $\mathrm{pre}_k(u)$ and $\mathrm{suf}_k(u)$, respectively, the prefix and the suffix of length k of u. Likewise, denote by $\mathrm{int}_k(u)$ the set of all proper subwords of length k of u. A language $L \subseteq X^+$ is said to be *k-testable* if and only if, for any strings $u, v \in X^+$, the equalities $\mathrm{pre}_k(u) = \mathrm{pre}_k(v)$, $\mathrm{suf}_k(u) = \mathrm{suf}_k(v)$, and $\mathrm{int}_k(u) = \mathrm{int}_k(v)$

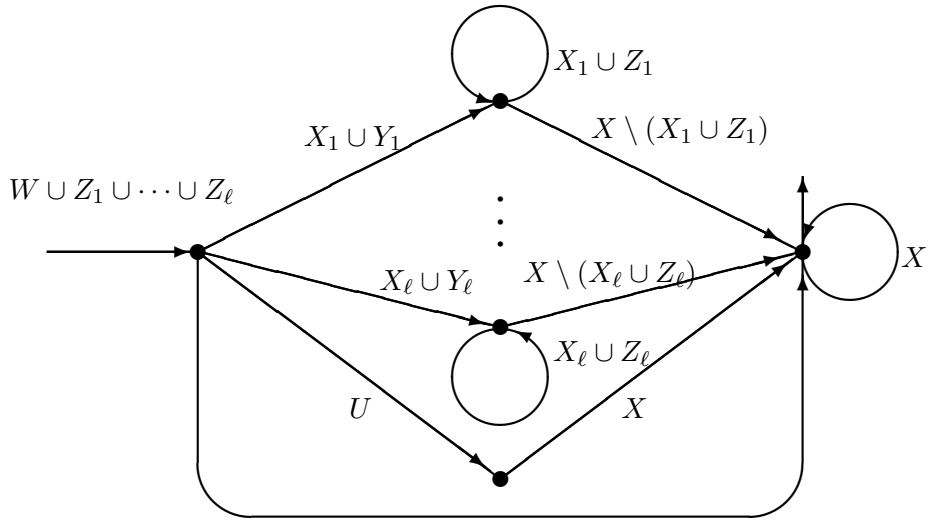

Figure 3: Alphabet and all sets of letters as in Fig. 2.

imply that $u \in L$ if and only if $v \in L$. A language is said to be *locally testable* if it is k-testable for some integer $k \geq 1$.

Theorem 20 ([24]) *All languages recognized by finite state automata* $\mathrm{Atm}(D, T, f, \ell)$ *with* $\mathrm{Im}(f) = \{-\}$ *are* 1*-testable.*

8 A new method for classification of strings

Let us define a classifier $\mathrm{CL}(V, E, \ell, r)$ as a quadruple

$$\mathrm{CL}(V, E, \ell, r) = (V, E, \ell, r), \tag{7}$$

where $V = \{v_1, \ldots, v_n\}$ is the set of vertices and E is the set of edges of a graph $G = (V, E)$ with multiple edges allowed and with each edge e labeled by a letter $\ell(e)$ of the alphabet X and a real number $r(e)$. In other words, there are two functions

$$\ell : E \to X \text{ and } r : E \to \mathbb{R}. \tag{8}$$

The *state* (or *current state*) of the classifier $\mathrm{CL}(V, E, \ell, r)$ is a labeling of all vertices by real numbers, i.e., a function

$$s : V \to \mathbb{R}. \tag{9}$$

Potentially the classifiers $\mathrm{CL}(V, E, \ell, r)$ can be used for both classification and clustering. A classification of any given set of DNA sequences is a partition of these sequences into several classes. Classifiers obtain classifications via various algorithms for supervised learning. In this way the classification is known for the given set of data. The problem is to construct a classifier that will produce this classification, so that it can then be used to determine class membership of new sequences. Initial partition is usually communicated by a supervisor to a machine learning process constructing the classifier. A different problem is that of clustering data. It deals with dividing a set of given sequences into classes not known

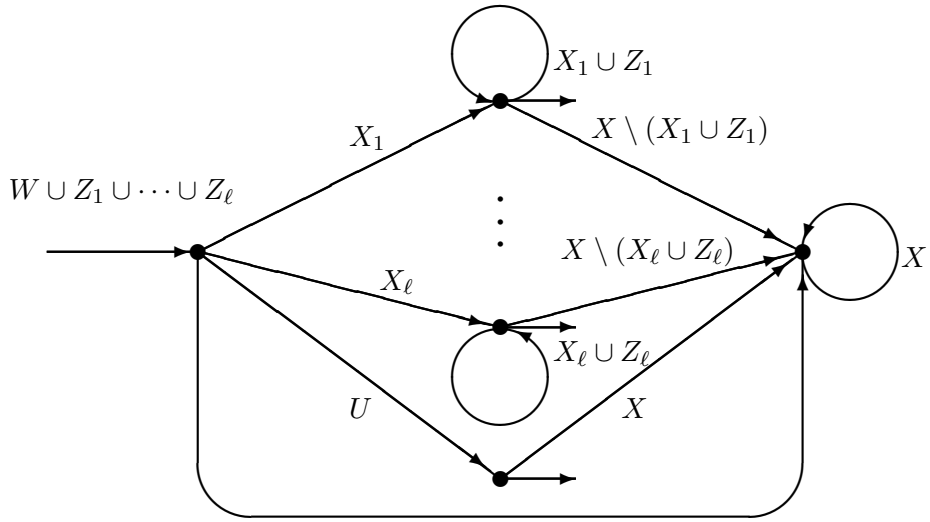

Figure 4: Alphabet $X = X_1 \cup \cdots \cup X_\ell \cup Z_1 \cup \cdots \cup Z_\ell \cup U \cup W$, where $X_i = \{x_{i1}, \ldots, x_{ik_i}\}, Z_i = \{z_{i1}, \ldots, z_{in_i}\}, U = \{u_1, \ldots, u_p\}, W = \{w_1, \ldots, w_t\}$, $p, \ell, t, n_1, \ldots, n_\ell \geq 0; \ell + p \geq 1; k_1, \ldots, k_\ell \geq 1$.

initially, but determined according to certain measures of similarities between sequences. This is usually accomplished via a process of unsupervised learning, see [49].

Now suppose that we want to use a classifier $\mathrm{CL}(V, E, \ell, r)$ to analyze a string

$$u = x_1, x_2, \ldots, x_N, \tag{10}$$

where $x_1, \ldots, x_N \in X$. The initial state $s_0 : V \to \mathbb{R}$ can be chosen arbitrarily depending on practical implementation. Then we use the labeled graph to recursively process all letters of the sequence u and modify the state of the graph. Suppose that after we have considered the first $i \geq 0$ letters of u the state of the graph is

$$s_i : V \to \mathbb{R}.$$

Then we can determine the next state s_{i+1} with recursion

$$s_{i+1}(v) = \sum_{w \in V, (w,v) \in E} r((w, v)) s_i(w). \tag{11}$$

After the whole sequence u has been processed, for every vertex $v \in V$, we know the final value $s_N(v) \in \mathbb{R}$.

Let us now define the standard partitions which we are going to use in classification of DNA sequences. The following standard partitions will be associated with the classifier $\mathrm{CL}(V, E, \ell, r)$. For every $1 \leq k \leq N$, we define the classification $\mathcal{K}_k$ as the one which divides all given DNA sequences into classes $C_1, \ldots, C_k$, by including the sequence u into the class $C_i = C_i^{(k)}$, where i is chosen so that $1 \leq i \leq k$, and

$$s_N(v_i) = \max\{s_N(v_1), \ldots, s_N(v_k)\}.$$

Obviously, for $k > 1$, every classification $\mathcal{K}_k$ can be obtained from $\mathcal{K}_{k-1}$ by selecting certain elements in all classes

$$C_1^{(k-1)}, C_2^{(k-1)}, \ldots, C_{k-1}^{(k-1)}$$

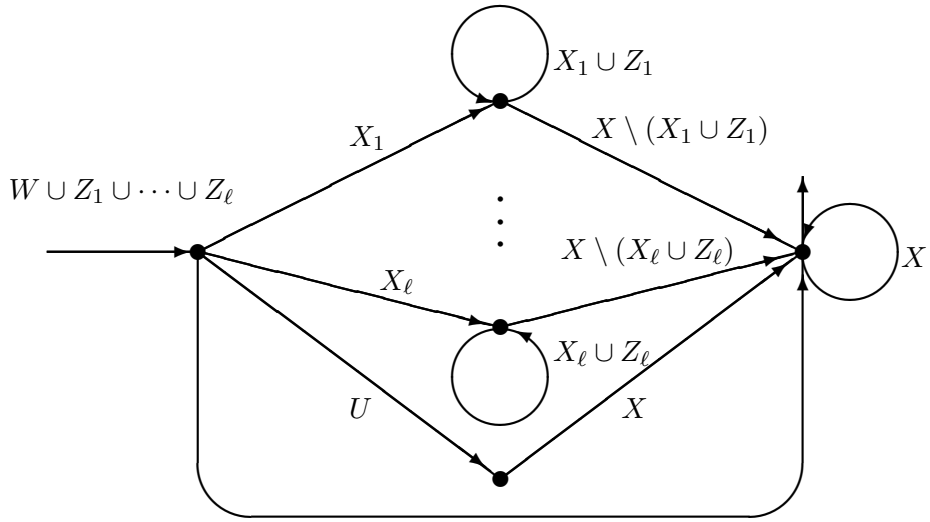

Figure 5: Alphabet and all sets of letters as in Fig. 4.

of $\mathcal{K}_{k-1}$ and including them in the new class $C_k^{(k)}$. Thus, every previous classification can be regarded as a simplified version of the next one, and every next classification is a refinement of the preceding one.

This new concept has been introduced in [19]. Our intention is to attract the attention of other researchers to this interesting and promising new notion. We are planning to investigate this construction further in the framework of our work on IRGS grant allocated by the University of Tasmania for the study of combinatorial algorithms for classifying DNA data collected by the School of Plant Science and CRC for Sustainable Production Forestry.

It is always essential to verify that a new model is suitable for handling sufficiently general classes of problems.

Proposition 2 ([19]) *For each set S of DNA sequences and every given partition*

$$S = S_1 \dot{\cup} S_2 \dot{\cup} \cdots \dot{\cup} S_k \tag{12}$$

one can find a classifier $\mathrm{CL}(V, E, \ell, r)$

$$C = \mathrm{CL}(V, E, \ell, r) = (V, E, \ell, r) \tag{13}$$

which produces classification

$$\mathcal{K} : X^* = C_1 \dot{\cup} C_2 \dot{\cup} \cdots \dot{\cup} C_k \tag{14}$$

such that the classes of partition (12) *are determined by the classes of classification* (14) *so that $S_i = S \cap C_i$ for all $i = 1, \ldots, k$.*

Thus, the classifiers $\mathrm{CL}(V, E, \ell, r)$ can handle all classification problems for DNA datasets given sufficient computing time.

This new model has some similarities with the concept of a finite state automaton and that of a neural network, but is different from them. The notion of a classifier $\mathrm{CL}(V, E, \ell, r)$

has been carefully chosen from the very beginning to combine the generality of finite state automata and the flexibility of neural networks.

Neural networks can be represented with similar labeled graphs. In this case the vertices are called neurons, and the labels of the edges are called weights. Edge labels are modified while a neural network is being trained. After that during the operation of the network the labels remain unchanged. Each neuron of the network takes a weighted sum of its inputs and passes it through a threshold function, usually the sigmoid function. The classifiers $\mathrm{CL}(V, E, \ell, r)$ are different from neural networks and finite state automata.

The major difference is that neural networks and the classifiers $\mathrm{CL}(V, E, \ell, r)$ are designed to solve substantially different types of problems. Neural networks cannot be directly applied to classification of DNA sequences without collections of some additional data, for example, from microarrays. The reason for this is that the operation of every neural network depends on a relatively small number of input parameters, represented as continuous real values. Small changes to the values of these parameters are not generally supposed to create changes to the classification outcome. Hence it is impossible to encode whole long DNA sequences in this way. In contrast, classifiers $\mathrm{CL}(V, E, \ell, r)$ can process all base pairs of a given DNA sequence in succession.

Sophisticated continuous threshold functions used in neural networks lead to another serious difference (see [31], Section 11). Although the current state of a classifier $\mathrm{CL}(V, E, \ell, r)$ appears similar to the state of a neural network, the transition to the next state is accomplished in a completely different fashion.

In conclusion let us briefly review how strings occur in analysis of DNA sequences. Recall that Watson and Crick discovered the structure of DNA in 1953. DNA molecule is a double helix consisting of two strands. Each strand is a string of 4 nucleotides or bases: A (adenine), C (cytosine), G (guanine), and T (thymine). According to the Watson-Crick complementarity each nucleotide in one strand is crosslinked to a complementary nucleotide in another strand, and together they form a base pair. In each DNA molecule, A and T always complement each other: A in one strand is linked to T in the second spiral. Similarly, C and G complement each other. Therefore the sequence of base pairs in every DNA molecule can be represented with just one string of letters A,C,G,T. Thus, the problem of classifying DNA molecules is equivalent to that of classifying strings of letters over the alphabet $X = \{A, C, G, T\}$.

9 Open Questions

Problem 1 ([20]) *Describe all sets of strings accepted by the automata* $\mathrm{Atm}(D, T, f, \ell)$ *with the graph labeling satisfying known combinatorial properties, see [13].*

Problem 2 ([20]) *For each automaton* $\mathrm{Atm}(D, T, f, \ell)$*, find a regular expression describing the set of strings recognized by the automaton.*

Problem 3 ([22]) *For each automaton* $\mathrm{Atm}(D, T, f, \ell)$*, describe all grammars generating sets of strings recognized by the automaton.*

Problem 4 ([20]) *Find conditions necessary and sufficient for a class of sets of strings recognised by automata* $\mathrm{Atm}(D, T, f, \ell)$ *of bipartite graphs (or other important special types of graphs) is closed under intersection, union, complement, concatenation and the Kleene star operation.*

In the special case, where each current state of the classifier $\mathrm{CL}(V, E, \ell, r)$ is a characteristic function of one of the vertices, the classifier can be regarded as an FSA, and a standard minimization algorithm applies (see [12]).

Problem 5 *Develop minimization algorithms for the general classifiers* $\mathrm{CL}(V, E, \ell, r)$ *with arbitrary current state functions.*

References

[1] Aho, A.V., Hopcroft, J. and Ullman, J.D., "The Design and Analysis of Computer Algorithms", Addison-Wesley, 1974.

[2] Bača, M., Jendroľ, S., Miller, M. and Ryan, J., *Antimagic labelings of generalized Petersen graphs that are plane*, Ars Combin. 73 (2004), 115–128.

[3] Bača, M. and Miller, M., *On d-antimagic labelings of type* $(1, 1, 1)$ *for prisms*, J. Combin. Math. Combin. Comput. 44 (2003), 199–207.

[4] Bača, M., Miller, M. and Slamin, *Vertex-magic total labelings of generalized Petersen graphs*, 11th Australasian Workshop on Combinatorial Algorithms (Hunter Valley, 2000). Int. J. Comput. Math. 79 (2002), no. 12, 1259–1263.

[5] Berstel, J., *Finite automata and rational languages: an introduction*, in "Formal Properties of Finite Automata and Applications", Lect. Notes Comp. Sci. 386, Springer, New York, 1989, 2–14.

[6] Berry, G. and Sethi, R., *From regular expressions to deterministic automata*, Theoret. Comp. Sci. 48 (1986), 117–126.

[7] Brüggeman-Klein, A., *Regular expressions into finite automata*, Theoret. Comp. Sci. 120 (1993), 197–213.

[8] Chang, C.H. and Paige, R., *From regular expressions to DFA's using compressed NFA's*, Proc. Third Symposium on Combinatorial Pattern Matching (1992), 90–110.

[9] Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C., "Introduction to Algorithms", The MIT Press, Cambridge, 2001.

[10] Dazeley, R.P. and Kang, B.H., *Weighted MCRDR: deriving information about relationships between classifications in MCRDR*, AI 2003: Advances in Artificial Intelligence, Perth, Australia, 2003, 245–255.

[11] Dazeley, R.P. and Kang, B.H., *An online classification and prediction hybrid system for knowledge discovery in databases*, Proc. AISAT 2004, The 2nd Internat. Conf. Artificial Intelligence in Science and Technology, Hobart, Tasmania, 2004, 114–119.

[12] Eilenberg, S., "Automata, Languages, and Machines", Vol. A,B, Academic Press, New York, 1974.

[13] Gallian, J.A., *Graph labeling*, Electronic J. Combinatorics, Dynamic Survey DS6, January 20, 2005, 148pp, www.combinatorics.org/surveys

[14] Gusfield, D., "Algorithms on Strings, Trees, and Sequences", Computer Science and Computational Biology, Cambridge University Press, Cambridge, 1997.

[15] Holub, J., Iliopoulos, C.S., Melichar, B. and Mouchard, L., *Distributed string matching using finite automata*, "Combinatorial Algorithms", AWOCA 99, Perth, 114–127.

[16] Kang, B.H., "Pacific Knowledge Acquisition Workshop". Auckland, New Zealand, 2004.

[17] Kelarev, A.V., "Ring Constructions and Applications", World Scientific, 2002.

[18] Kelarev, A.V., "Graph Algebras and Automata", Marcel Dekker, 2003.

[19] Kelarev, A.V., Kang, B.H., Sale, A.H.J. and Williams, R.N., *Labeled graphs as classifiers for DNA code*, School of Computing, University of Tasmania, Technical Report, 22 April 2006, 10pp.

[20] Kelarev, A.V., Miller, M. and Sokratova, O.V., *Directed graphs and closure properties for languages*, "Proc.12 Australasian Workshop on Combinatorial Algorithms" (Ed. E.T. Baskoro), Putri Gunung Hotel, Lembang, Bandung, Indonesia, July 14–17, 2001, pp.118–125. [F1]

[21] Kelarev, A.V., Miller, M. and Sokratova, O.V., *Languages recognized by two-sided automata of graphs*, Proc. Estonian Akademy of Science 54 (2005) (1), 46–54.

[22] Kelarev, A.V. and Sokratova, O.V., *Languages recognized by a class of finite automata*, Acta Cybernetica 15 (2001), 45–52.

[23] Kelarev, A.V. and Sokratova, O.V., *Directed graphs and syntactic algebras of tree languages*, J. Automata, Languages & Combinatorics 6 (2001)(3), 305–311.

[24] Kelarev, A.V. and Sokratova, O.V., *Two algorithms for languages recognized by graph algebras*, Internat. J. Computer Math. 79 (2002)(12) 1317–1327.

[25] Kelarev, A.V. and Sokratova, O.V., *On congruences of automata defined by directed graphs*, Theoret. Computer Science 301 (2003), 31–43.

[26] Kelarev, A.V. and Trotter, P.G., *A combinatorial property of automata, languages and their syntactic monoids*, Proceedings of the Internat. Conf. Words, Languages and Combinatorics III, Kyoto, Japan, 2003, 228–239.

[27] Lallement, G., "Semigroups and Combinatorial Applications", Wiley, New York, 1979.

[28] Lee, K.H., Kang, B.H., *A new framework for uncertainty sampling: exploiting uncertain and positive-certain examples in similarity-based text classification*, Proc. Internat. Conf. on Information Technology: Coding and Computing (ITCC2004), Las Vegas, Nevada, 2004, 12pp.

[29] Lin, Y. and Miller, M., *Vertex magic total labelings of complete graphs*, Bull. Inst. Combin. Appl. 33 (2001), 68–76.

[30] Lin, Y., Slamin, Bača, M. and Miller, M., *On d-antimagic labelings of prisms*, Ars Combin. 72 (2004), 65–76.

[31] Luger, G.F, "Artificial Intelligence. Structures and Strategies for Complex Problem Solving" Addison-Wesley, 2005.

[32] Miller, M., Patel, D., Ryan, J., Sugeng, K.A., Slamin and Tuga, M., *Exclusive sum labeling of graphs*, J. Combin. Math. Combin. Comput. 55 (2005), 137–148.

[33] Miller, M., Rodger, C. and Simanjuntak, R., *Distance magic labelings of graphs*, Australas. J. Combin. 28 (2003), 305–315.

[34] Oates-Williams, S., *Graphs and universal algebras, In: "Combinatorial Mathematics", VIII (Geelong, 1980)*, Springer, Berlin, 1981, 351–354.

[35] Park, S.S., Kim, Y., Park, G., Kang, B.H., Compton, P., *Automated information mediator for HTML and XML Based Web information delivery service*, Proc. 18th Australian Joint Conf. on Artificial Intelligence , Sydney, 2005, 401–404.

[36] Păun, G. and Salomaa, A., "New Trends in Formal Languages", Springer-Verlag, Berlin, 1997.

[37] Petrovskiy, M.: Probability estimation in error correcting output coding framework using game theory. AI 2005: Advances in Artificial Intelligence, Sydney, Australia, 2005, Lect. Notes Artificial Intelligence 3809 (2005) 186–196.

[38] Pin, J.E., "Formal Properties of Finite Automata and Applications", Lect. Notes Computer Science 386, Springer, New York, 1989.

[39] Rozenberg, G. and Salomaa, A., "Handbook of Formal Languages", Vol. 1, Word, Language, Grammar, Springer-Verlag, Berlin, 1997.

[40] Slamin, Bača, M., Lin, Y., Miller, M. and Simanjuntak, R., *Edge-magic total labelings of wheels, fans and friendship graphs*, Bull. Inst. Combin. Appl. 35 (2002), 89–98.

[41] Smyth, B., "Computing Patterns in Strings", Addison-Wesley, 2003.

[42] Street, A.P. and Wallis, W.D., "Combinatorial Theory: An Introduction", Charles Babbage Research Centre, 1977.

[43] Sugeng, K.A., Miller, M., Lin, Y. and Bača, M., *Super* (a, d)*-vertex-antimagic total labelings*, J. Combin. Math. Combin. Comput. 55 (2005), 91–102.

[44] Sugeng, K.A. and Miller, M., *Relationship between adjacency matrices and super* (a, d)*-edge-antimagic-total labeling of graphs*, J. Combin. Math. Combin. Comput. 55 (2005), 71–82.

[45] Sugeng, K.A., Miller, M., Slamin and Bača, M., (a, d)*-edge-antimagic total labelings of caterpillars*, Lecture Notes in Comput. Sci. 3330 (2005), 169–180.

[46] Thompson, K., *Regular expressions search algorithm*, Comm. ACM 11 (1968), 410–422.

[47] Tuga, M., Miller, M., Ryan, J. and Ryjáček, Z., *Exclusive sum labelings of trees*, J. Combin. Math. Combin. Comput. 55 (2005), 109–121.

[48] Tuga, M. and Miller, M., Δ-*optimum exclusive sum labeling of certain graphs with radius one*, Lecture Notes in Comput. Sci. 3330 (2005), 216–225.

[49] Witten, I.H. and Frank, E., "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations", Morgan Kaufmann, 2005.

[50] van Leeuwen, J., "Handbook of Theoretical Computer Science", Vol. A,B, Algorithms and Complexity, Elsevier, Amsterdam, 1990.

[51] Yu, S., *Regular Languages*, "Handbook of Formal Languages", Vol. 1, Springer, New York (1997), 41–110.